Fondamenti di Informatica T-1, 2019/2020 – Modulo 2

Prova d'Esame 2A di Giovedì 23 Gennaio 2020 – tempo a disposizione 2h

Avvertenze per la consegna: apporre all'inizio di ogni file sorgente un commento contenente i propri dati (**cognome, nome, numero di matricola**) e il **numero** della prova d'esame. Al termine, **consegnare tutti i file sorgenti** necessari alla compilazione e alla corretta esecuzione del programma.
Nota: **il main non è opzionale**; i *test* richiesti vanno implementati.
Consiglio: per verificare l'assenza di *warning*, eseguire sempre *"Rebuild All"*.

Una banca eroga prestiti a privati cittadini, che poi restituiscono nel tempo il denaro. Al fine di tenere traccia di quanto prestato, la banca scrive su un file di testo le informazioni relative a tali prestiti. In particolare, per ogni prestito effettuato, la banca scrive su ogni riga: **l'identificatore** unico della pratica (un intero); a seguire, seguito da uno spazio, il **cognome** della persona a cui ha effettuato il prestito (una stringa di 2047 caratteri utili, contenente spazi); a seguire, separato dal carattere ';', il **nome** della persona a cui ha effettuato il prestito (una stringa di 2047 caratteri utili, contenente spazi); infine, separato ancora da un carattere ';', **l'importo** del prestito (un float). Non è noto a priori quanti prestiti siano memorizzati nel file, ed è possibile che una stessa persona abbia più prestiti con la stessa banca. Inoltre, quando un utente ripaga per intero il suo prestito (e quindi lo estingue), gli impiegati della banca hanno la prassi di non cancellare la riga corrispondente, bensì modificano il cognome della persona e vi aggiungono in testa la stringa "ESTINTO". Si veda a titolo di esempio il file di testo "**`prestiti.txt`**" fornito nello StartKit.

Esercizio 1 – Struttura dati Loan, e funzioni di lett./scritt. (mod. element.h e banca.h/.c)

Si definisca una struttura dati **`Loan`** per memorizzare i dati relativi ad un prestito, cioè l'id della pratica, cognome e nome della persona, e importo del prestito.

Si definisca la funzione:

```
Loan leggiUnLoan(FILE * fp);
```

che, ricevuto in ingresso un puntatore ad una struttura dati di tipo **`FILE`**, contenente le informazioni riguardo i prestiti, legga i dati relativi ad un singolo prestito e li restituisca tramite una struttura dati di tipo **`Loan`**. Qualora vi siano problemi nella lettura, la funzione restituisca una struttura dati con id del **`Loan`** pari a zero.

Si definisca la funzione:

```
Loan * leggiLoanAttivi(char * fileName, int * dim);
```

che, ricevuto in ingresso il nome di un file che contenga i prestiti, legga da tale file i dati e li restituisca tramite un vettore di strutture dati di tipo **`Loan`**, allocato dinamicamente della dimensione minima necessaria. Tramite il parametro **`dim`** passato per riferimento, la funzione deve restituire la dimensione del vettore. Qualora la funzione incontri dei problemi nella lettura, o vi siano errori nell'apertura del file, la funzione dovrà stampare un messaggio di errore a video, restituire un puntatore a **`NULL`** e **`dim`** pari a zero. Si presti attenzione però: il file dei prestiti contiene anche i prestiti già estinti, ma la funzione dovrà restituire solo ed unicamente i prestiti ancora attivi. A tal scopo, si ricorda l'esistenza della funzione

```
char *strstr(const char * bigString, const char * target);
```

che cerca la stringa **`target`** (più piccola) nella stringa (più grande) **`bigString`**, e restituisce un puntatore alla prima occorrenza di **`target`** (se presente) o un puntatore a **`NULL`** se **`target`** non compare in **`bigString`**.

Il candidato abbia cura di realizzare nel main opportuni test al fine di verificare il corretto funzionamento delle funzioni di cui sopra, avendo cura di deallocare la memoria, se necessario.

Fondamenti di Informatica T-1, 2019/2020 – Modulo 2

Prova d'Esame 2A di Giovedì 23 Gennaio 2020 – tempo a disposizione 2h

Esercizio 2 –Ordinamento ed estrazione per chiave (moduli element.h/.c e banca.h/.c)

Si definisca la procedura:

```
void ordina(Loan * elenco, int dim);
```

che, ricevuto in ingresso un array di strutture dati di tipo **`Loan`**, e la sua dimensione dim, ordini il vetorre secondo il seguente criterio: prima in ordine alfabetico in base al cognome; a parità di cognome, in base al nome; infine, a parità di nome, in ordine decrescente in base all'importo (dal più grande al più piccolo). Per ordinare il candidato usi l'algoritmo "insert sort" visto a lezione.

Gli impiegati della banca hanno bisogno di sapere tutti i prestiti fatti ad una certa persona. A tal scopo, il candidato definisca una funzione:

```
list estrai(Loan * elenco, int dim, char * cognome, char * nome)
```

che, ricevuti in ingresso un array di strutture dati di tipo **`Loan`** e la dimensione **`dim`**, ed una stringa **`cognome`** ed una stringa **`nome`** indicanti l'individuo, restituisca in uscita una lista di **`Loan`**. La lista dovrà contenere in particolare tutti i prestiti l'individuo specificato.

Esercizio 3 – Verifica dell'esposizione totale verso un singolo individuo (modulo banca.h/banca.c)

Una delle truffe più frequenti consiste in un solo, singolo individuo che richiede più prestiti contemporaneamente alla stessa banca (magari rivolgendosi a diverse filiali). A tal scopo, la banca in ogni momento deve conoscere l'esposizione totale verso ogni singolo individuo. A tal scopo, si sviluppi una procedura:

```
void espo(Loan * elenco, int dim);
```

che, ricevuti in ingresso l'array **`elenco`** dei **`Loan`** correntemente attivi e la dimensione di tale array, stampi a video l'esposizione totale verso ogni singolo individuo.
L'esposizione totale verso un individuo è calcolata come la somma di tutti i prestiti effettuati a quell'individuo. Ogni individuo è identificato univocamente dal proprio cognome e nome (non esistono cioè casi di omonimia). L'esposizione totale verso ogni individuo dovrà essere stampata una e una sola volta per ogni individuo.

Esercizio 4 Stampa dei risultati, e de-allocazione memoria (main.c)

Il candidato realizzi nella funzione **`main(…)`** un programma che stampi a video, utilizzando le funzioni di cui ai punti precedenti, l'esposizione totale verso ogni singolo individuo.
Al termine del programma, il candidato abbia cura di de-allocare tutta la memoria allocata dinamicamente, ivi compresa la memoria allocata per le liste.

Fondamenti di Informatica T-1, 2019/2020 – Modulo 2

Prova d'Esame 2A di Giovedì 23 Gennaio 2020 – tempo a disposizione 2h

"element.h":

```
#define _CRT_SECURE_NO_WARNINGS

#pragma once

#ifndef _ELEMENT_H
#define _ELEMENT_H

#include <stdio.h>
#include <string.h>

#define DIM_COGNOME 2048
#define DIM_NOME 2048

typedef struct {
	int id;
	char cognome[DIM_COGNOME];
	char nome[DIM_NOME];
	float importo;
} Loan;

typedef Loan element;

int compare(Loan l1, Loan l2);

#endif
```

"element.c":

```
#include "element.h"

int compare(Loan l1, Loan l2) {
	int result;
	result = strcmp(l1.cognome, l2.cognome);
	if (result == 0)
		result = strcmp(l1.nome, l2.nome);
	if (result == 0) {
		if (l1.importo > l2.importo)
			result = -1;
		else
			if (l1.importo < l2.importo)
				result = 1;
			else
				result = 0;
	}
	return result;
}
```

Fondamenti di Informatica T-1, 2019/2020 – Modulo 2

Prova d'Esame 2A di Giovedì 23 Gennaio 2020 – tempo a disposizione 2h

"list.h"

```
#ifndef LIST_H
#define LIST_H

#include "element.h"

typedef struct list_element {
    element value;
    struct list_element  *next;
} item;

typedef  item* list;

typedef  int boolean;

/* PRIMITIVE  */
list emptylist(void);
boolean empty(list);
list cons(element, list);
element head(list);
list tail(list);

void showlist(list l);
void freelist(list l);

#endif
```

"list.c":

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include "list.h"

/* OPERAZIONI PRIMITIVE */
list  emptylist(void)            /* costruttore lista vuota */
{
    return NULL;
}

boolean  empty(list l)     /* verifica se lista vuota */
{
    return (l==NULL);
}

list  cons(element e, list l)
{
  list t;       /* costruttore che aggiunge in testa alla lista */
  t=(list)malloc(sizeof(item));
  t->value=e;
  t->next=l;
  return(t);
}

element head(list l) /* selettore testa lista */
{
```

```
  if (empty(l)) exit(-2);
  else return (l->value);
}

list  tail(list l)         /* selettore coda lista */
{
  if (empty(l)) exit(-1);
  else return (l->next);
}

void showlist(list l) {
       element temp;
       if (!empty(l)) {
              temp = head(l);
              printf("%d %s %s %.2f\n",
                     temp.id, temp.cognome, temp.nome, temp.importo);
              showlist(tail(l));
              return;
       }
       else {
              printf("\n\n");
              return;
       }
}

void freelist(list l) {
       if (empty(l))
              return;
       else {
              freelist(tail(l));
              free(l);
       }
       return;
}
```

Fondamenti di Informatica T-1, 2019/2020 – Modulo 2

Prova d'Esame 2A di Giovedì 23 Gennaio 2020 – tempo a disposizione 2h

"banca.h":

```
#pragma once

#ifndef _BANCA_H
#define _BANCA_H

#include "element.h"
#include "list.h"

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

// Es. 1
Loan leggiUnLoan(FILE * fp);
Loan * leggiLoanAttivi(char * fileName, int * dim);

// Es. 2
void ordina(Loan * elenco, int dim);
list estrai(Loan * elenco, int dim, char * cognome, char * nome);

// Es. 3
void espo(Loan * elenco, int dim);

#endif
```

"banca.c":

```
#include "banca.h"


Loan leggiUnLoan(FILE * fp) {
	Loan result;
	char ch;
	int i;

	if (fscanf(fp, "%d", &(result.id)) == 1) {
		fgetc(fp);
		i = 0;
		do {
			ch = fgetc(fp);
			if (ch != ';' && i < DIM_COGNOME) {
				result.cognome[i++] = ch;
			}
		} while (ch != ';' && i < DIM_COGNOME);
		result.cognome[i] = '\0';
		i = 0;
		do {
			ch = fgetc(fp);
			if (ch != ';' && i < DIM_NOME) {
				result.nome[i++] = ch;
			}
		} while (ch != ';' && i < DIM_NOME);
		result.nome[i] = '\0';
		fscanf(fp, "%f", &(result.importo));
	}
```

```
	else {
		result.id = 0;
	}
	return result;
}

Loan * leggiLoanAttivi(char * fileName, int * dim) {
	FILE * fp;
	int counter;
	Loan * result = NULL;
	Loan temp;

	fp = fopen(fileName, "rt");
	if (fp != NULL) {
		counter = 0;
		do {
			temp = leggiUnLoan(fp);
			if (temp.id != 0 && strstr(temp.cognome, "ESTINTO") == NULL)
				counter++;
		} while (temp.id != 0);
		if ( counter > 0) {
			rewind(fp);
			result = (Loan *)malloc(sizeof(Loan) * counter);
			*dim = 0;
			do {
				temp = leggiUnLoan(fp);
				if (temp.id != 0 && strstr(temp.cognome, "ESTINTO") == NULL) {
					result[*dim] = temp;
					(*dim)++;
				}
			} while (temp.id != 0);
		}
		fclose(fp);
	}
	else {
		result = NULL;
		*dim = 0;
	}
	return result;
}



void insOrd(Loan v[], int pos) {
	int i = pos - 1;
	Loan x = v[pos];
	while (i >= 0 && compare(x,v[i])<0) {
		v[i + 1] = v[i]; /* crea lo spazio */
		i--;
	}
	v[i + 1] = x;/* inserisce l'elemento */
}

void insertSort(Loan v[], int n) {
	int k;
	for (k = 1; k < n; k++)
		insOrd(v, k);
```

```
}

void ordina(Loan * elenco, int dim) {
	insertSort(elenco, dim);
}

list estrai(Loan * elenco, int dim, char * cognome, char * nome) {
	list result;
	int i;

	result = emptylist();
	for (i = 0; i < dim; i++) {
		if (strcmp(elenco[i].cognome, cognome) == 0
			&& strcmp(elenco[i].nome, nome) == 0) {
			result = cons(elenco[i], result);
		}
	}
	return result;
}

int stessoIndividuo(Loan l1, Loan l2) {
	return strcmp(l1.cognome, l2.cognome) == 0
		&&
		strcmp(l1.nome, l2.nome) == 0;
}

void espo(Loan * elenco, int dim) {
	int i;
	list temp1, temp2;
	float amount;

	ordina(elenco, dim);

	for (i = 0; i < dim; i++) {
		if (i == 0 || !stessoIndividuo(elenco[i], elenco[i - 1]) ) {
			temp1 = estrai(elenco, dim, elenco[i].cognome, elenco[i].nome);
			temp2 = temp1;
			amount = 0;
			while (!empty(temp2)) {
				amount = amount + head(temp2).importo;
				temp2 = tail(temp2);
			}
			freelist(temp1);
			printf("Sig. %s %s: %.2f\n", elenco[i].cognome, elenco[i].nome,
amount);
		}
	}
}
```

Fondamenti di Informatica T-1, 2019/2020 – Modulo 2

Prova d'Esame 2A di Giovedì 23 Gennaio 2020 – tempo a disposizione 2h

"main.c":

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

#include "element.h"
#include "banca.h"


int main() {
	{ // Es. 1
		Loan * prestiti;
		int dim;
		int i;

		prestiti = leggiLoanAttivi("prestiti.txt", &dim);
		for (i = 0; i < dim; i++) {
			printf("%d %s %s: %f\n", prestiti[i].id, prestiti[i].cognome,
				prestiti[i].nome, prestiti[i].importo);
		}
		printf("\n\n");
		free(prestiti);
	}
	{ // Es. 2
		Loan * prestiti;
		list estratto;
		int dim;
		int i;

		prestiti = leggiLoanAttivi("prestiti.txt", &dim);
		ordina(prestiti, dim);
		for (i = 0; i < dim; i++) {
			printf("%d %s %s: %f\n", prestiti[i].id, prestiti[i].cognome,
				prestiti[i].nome, prestiti[i].importo);
		}
		printf("\n\n");
		estratto = estrai(prestiti, dim, "Chesani Mazzanti", "Federico Domenico
Maria");
		showlist(estratto);
		freelist(estratto);
		free(prestiti);

	}
	{ // Es. 3
		Loan * prestiti;
		int dim;


		prestiti = leggiLoanAttivi("prestiti.txt", &dim);
		ordina(prestiti, dim);
		espo(prestiti, dim);
		free(prestiti);

	}

	return 0;
}
```

Fondamenti di Informatica T-1, 2019/2020 – Modulo 2

Prova d'Esame 2A di Giovedì 23 Gennaio 2020 – tempo a disposizione 2h

"prestiti.txt":

```
34 Chesani Mazzanti;Federico Domenico Maria;340.00
45 ESTINTOChesani Mazzanti;Federico Domenico Maria;100.00
61 Mello;Paola;115.00
63 ESTINTOMello;Paola;61.00
70 ESTINTOChesani Mazzanti;Federico Domenico Maria;1680.00
73 Chesani Mazzanti;Federico Domenico Maria;3400.00
74 Chesani Mazzanti;Federico Domenico Maria;6600.00
```